# Jalios Delivery Engine 2.0 (JADE)

# Manuel d'installation et d'exploitation





# A propos de ce document

## Contacts

Jalios SA
58 rue Pottier
78150 Le Chesnay

Si vous avez des questions ou souhaitez des éclaircissements sur ce document, vous pouvez nous contacter :

### Service commercial

Jean-François Pellier

Tél : 01 39 23 31 15
E-mail : jean-francois.pellier@jalios.com

### Service technique

Xuan Tuong LE

Tél : 06 14 97 44 44
E-mail : xuan-tuong.le@jalios.com

## Table des matières

# 1. Introduction

Ce document décrit l'architecture, l'installation et la configuration de JADE (JAlios Delivery Engine). Il est destiné aux équipes de développement et d'exploitation des sites JCMS.
JADE est un environnement complet d'intégration continue pour Jalios JCMS. Cette plateforme intègre tous les composants nécessaires pour :

- Mesurer automatiquement la qualité de développement et en continu
- Superviser et détecter les problèmes le plus rapidement possible
- Produire automatiquement des livrables avec l'intelligence de construction assurée par Jalios
- Centraliser des informations communes

# 2. Prérequis

Les composants décrits dans cette section sont nécessaires au bon fonctionnement de JADE. Dans la suite de ce document, ils seront considérés comme installés et testés.

Composants obligatoires :

- Serveur VMware ESX
- Serveur SMTP

## Configuration du serveur

JADE nécessite d'être installée sur un équipement serveur correspondant à l'une des configurations suivante :

- HP Proliant ML350 G6 E5620 ou supérieur (tour)
- HP Proliant DL 160 E5-2620 Gen8 ou supérieur (rack)
- HP lame Integrity BL860ci4 ou supérieur (blade)
- Dell PowerEdge T110 II ou supérieur (tour)
- Dell PowerEdge R210 II ou supérieur (rack)
- Dell PowerEdge M620 ou supérieur (blade)

Le serveur doit disposer d'au moins 6 Go de RAM et 250 Go d'espace disque. Selon la volumétrie de données, il peut être nécessaire de disposer de plus mémoire et/ou d'espace.

Pour se familiariser avec la méthodologie de développement de JCMS, il est indispensable de consulter les documents suivants :

- JCMS - Développement orienté module avec Eclipse et SVN
- JCMS : Développer des tests unitaires avec JCMS

A noter que pour exploiter et maintenir JADE, les documents décrits ci-dessus ne sont pas nécessaires.

# 3. Architecture de JADE

JADE est livré sous forme de machine virtuelle ayant le format .ova (*Open Virtual Appliance Format*) et elle fonctionne au sein d'un serveur VMware ESX. Pour permettre de démarrer rapidement une plateforme d'intégration continue pour des projets JCMS, JADE contient principalement les composants suivants qui sont préinstallés :

1. Jenkins : serveur d'intégration continue
2. SVN : serveur de gestion de version
3. les scripts de build de JCMS pré-paramétrés pour produire des modules (*.zip*) et l'application (*.war*)

Lorsque la machine virtuelle démarre, JADE est prêt à utiliser.

# 4. Installation

Cette étape d'installation consiste simplement à déployer JADE sur un serveur ESX et à démarrer la VM. Procédez aux étapes suivantes pour installer JADE :

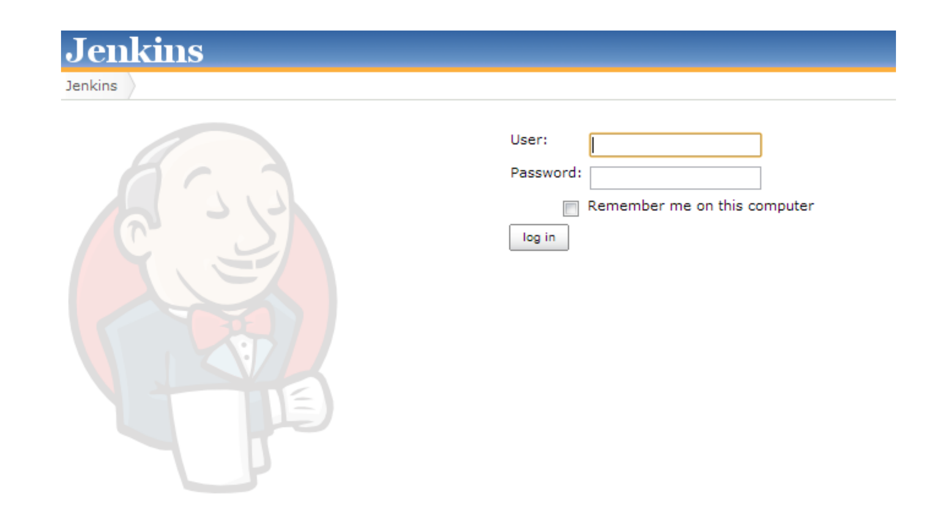
1. Récupérez JADE sous forme de fichier `.ova` disponible sur le site de support de Jalios
2. Ouvrez vSphere client
3. Choisissez « Fichier / Déployer modèle OVF »
4. Sélectionnez le fichier `jade.ova` et cliquer sur « Suivant »
5. Cliquez une nouvelle fois sur « Suivant »
6. Choisissez un nom et un emplacement pour la VM, puis cliquez sur « Suivant »
7. Choisissez l'hôte sur lequel la machine doit être déployée et cliquez sur « Suivant »
8. Choisissez le type de provisionnement pour le stockage de la VM, suivant les règles utilisées pour vos VMs :

    1. Le provisionnement « Thin provision » ou « provisionnement fin » permet d'avoir une VM qui n'utilise que l'espace nécessaire aux données. La création du disque est très rapide mais lorsque des données viennent s'ajouter, les performances d'accès peuvent en souffrir ;

    2. Le provisionnement statique immédiatement mis à zéro permet de réserver l'ensemble de l'espace alloué à la VM (100 Go au 28/02/2012). Tout le disque est mis à zéro (formatage), ce qui peut être long si le disque à créer est volumineux et/ou si la banque de stockage n'est pas très rapide

    3. Le provisionnement statique mis à zéro en différé : Création du disque et réservation d'une partie de l'espace pour écriture des données. Le reste du disque est mis à zéro quand la VM en a besoin. Cela permet de créer le disque très rapidement

9. Choisissez le réseau sur lequel la VM doit être connectée, puis cliquer sur « Suivant » ;
10. Vérifiez le résumé d'installation pour valider tous les choix effectués puis cliquez sur « Terminer ». Si vous désirez que la VM démarre immédiatement après son installation sur l'ESX, activez « Mettre sous tension après le déploiement »
11. Vérifiez que JADE démarre en allant à son URL via un navigateur : `http://jade.company.net/jenkins`. Dans l'exemple de 'URL, le nom d'hôte est `jade.company.net`. Ce nom est choisi par l'administrateur du réseau. Si la page suivante affiche, JADE est prête. Félicitations !

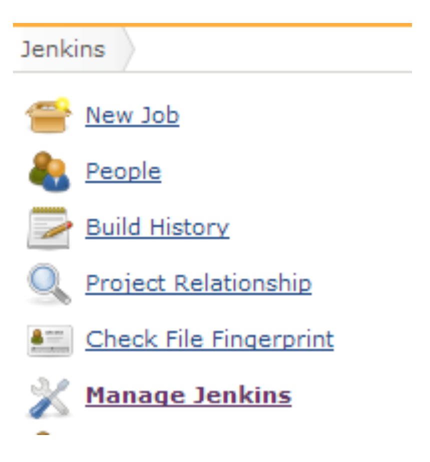

Jenkins
Jenkins
User:
Password:
Remember me on this computer
log in

# 5. Premiers pas avec JADE

## Configuration d'administrateur

**Objectif :** Sécuriser le compte d'administrateur de JADE
**Etapes :**

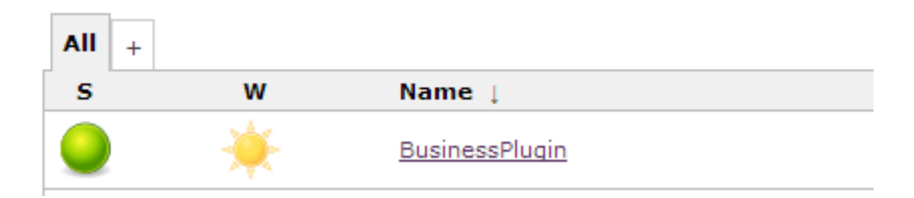
1. Connectez-vous sur `JADE`
2. Authentifiez-vous avec le compte d'administrateur `jade/jade`
3. Cliquez sur `Manage Jenkins` puis `Manager Users`
4. Editez l'utilisateur `jade` et générez un nouveau mot de passe sécurisé



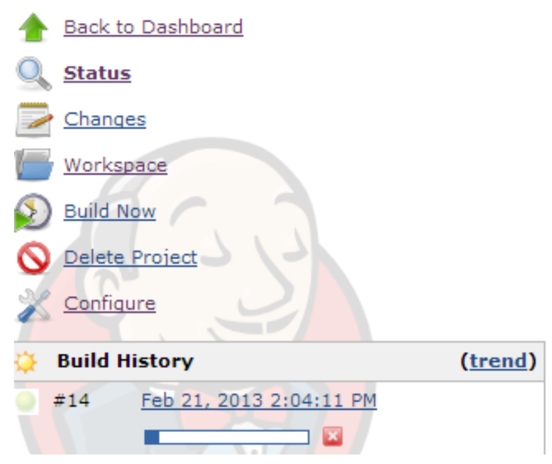

## Construction d'un module

**Objectif :** Construire le livrable du module qui est dans le format `.zip`
**Etapes :**

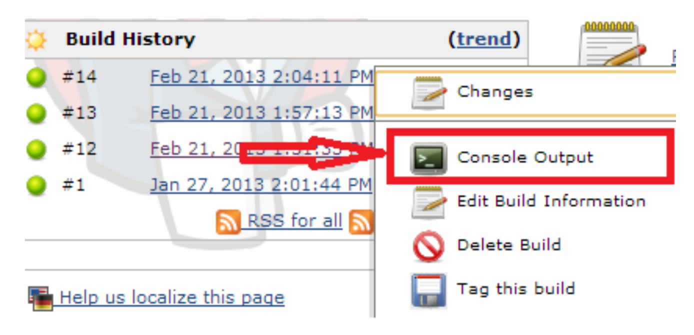
1. Connectez-vous sur `JADE`
2. Authentifiez-vous avec le compte d'administrateur `jade`
3. Entrez dans le projet `BusinessPlugin` servant de modèle de base pour vos futurs modules.




4. Cliquez sur `Build Now`. Le nouveau build démarre.



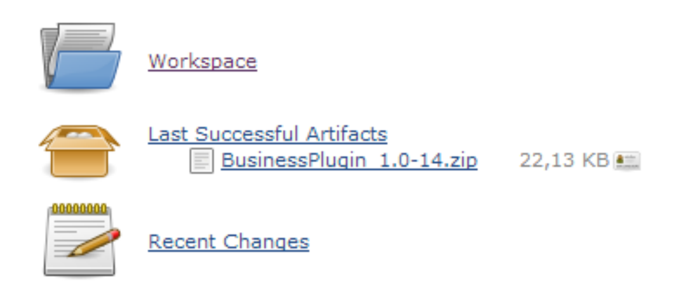

5. Jenkins permet de suivre le déroulement du build en mode interactif. Pour ce faire, cliquez sur `Console Output`.




6. Une fois le build terminé, l'icône de status est à jour :

   - vert : le build s'est correctement déroulé
   - jaune : le build s'est correctement déroulé mais les tests ont échoué. Il est possible de paramétrer JADE pour arrêter le build dans ce cas. La configuration de ce comportement sera abordée dans les sections ultérieures.
   - rouge : le build n'a pas abouti

7. Cliquez ensuite sur le lien du build pour accéder aux détails

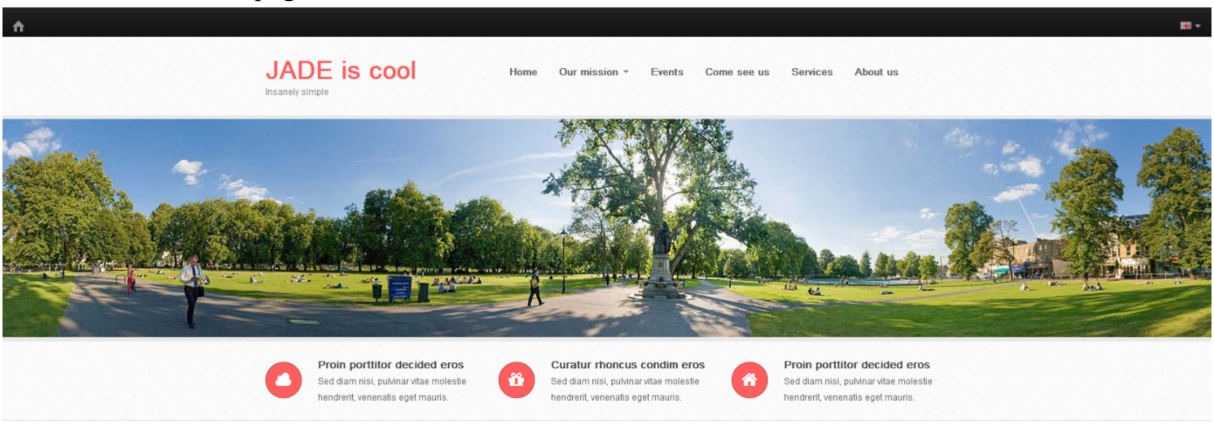


Vous venez de construire un module manuellement. Cette tâche peut être automatisable et déclenchée sur différents critères. Nous aborderons ce sujet dans les sections suivantes.

## Construction d'une webapp

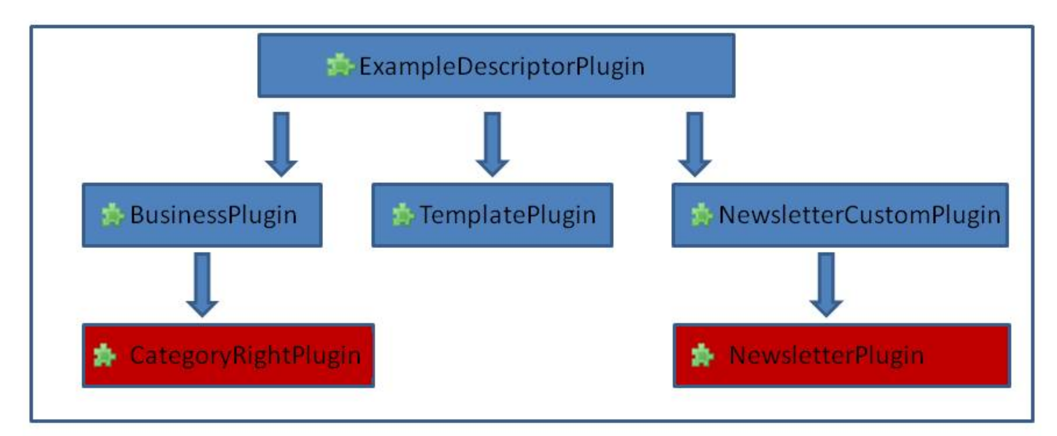
**Objectif :** Construire le war d'une webapp (`.war`)

**Description :**
Le livrable est une webapp JCMS construite sur une charte simple. Si vous déployez la webapp, voici le résultat de la page d'accueil.




Techniquement, cette webapp de démonstration est constituée de six modules. Les flèches bleues représentent la dépendance des modules :

- Les modules en rouge sont fournis par Jalios
- Ceux en bleu sont développés spécifiquement




ExampleWebapp

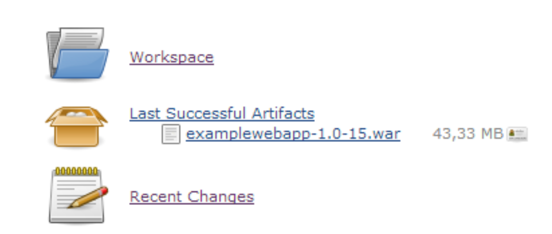
**Etapes :**

1. La procédure est la même que celle pour construire un module. Entrez dans le projet ExampleWebapp et cliquez sur `Build Now`. Seule la configuration du projet change.

2. Le résultat du build est le fichier `.war` visible dans la section `Last Successful Artifacts`

**Project ExampleWebapp**

ExampleWebapp




**Permalinks**

- Last build (#15), 29 days ago
- Last stable build (#15), 29 days ago
- Last successful build (#15), 29 days ago

## Résumé

Dans cette partie, nous avons abordé les sujets suivants :

- installer JADE rapidement
- construire un module
- construire une webapp contenant des modules

Dans la suite du document, nous allons détailler le mode de fonctionnement de JADE ainsi que les paramétrages pré-configurées à travers le projet `ExampleWebapp`.
Le but est de vous permettre de comprendre la configuration du projet `ExampleWebapp` afin de l'appliquer dans votre propre contexte selon les préconisations de Jalios.

# 6. Paramétrage de JADE

JADE est préconfigurée pour faciliter au maximum son utilisation. Néanmoins, certains paramètres sont nécessaires pour terminer l'installation de JADE dans votre environnement.

## Configurer l'URL de Jenkins

Procédez aux étapes suivantes pour configurer :

1. Connectez en tant qu'administrateur sur JADE
2. Allez dans `Manage Jenkins > Configure system`
3. Allez à `Jenkins location`
4. Remplissez les champs `Jenkins URL` et `System Admin e-mail address`. Pour plus de détails concernant ces paramètres, une explication détaillée est indiquée au niveau du champ.

**Jenkins Location**

| | |
|---|---|
| Jenkins URL | http://jade/ |
| System Admin e-mail address | noreply-jade@jalios.com |

## Configurer l'E-mail de notification

Procédez les étapes suivantes :

1. Connectez en tant qu'administrateur sur JADE
2. Allez dans `Manage Jenkins > Configure system`
3. Allez à `E-mail Notification`
4. Remplissez les champs `SMTP Server` et `Default user e-mail suffix`. Vous avez la possibilité de tester directement en cochant la case `Test configuration by sending test e-mail`

**E-mail Notification**

SMTP server

Default user e-mail suffix

☑ Test configuration by sending test e-mail

Test e-mail recipient

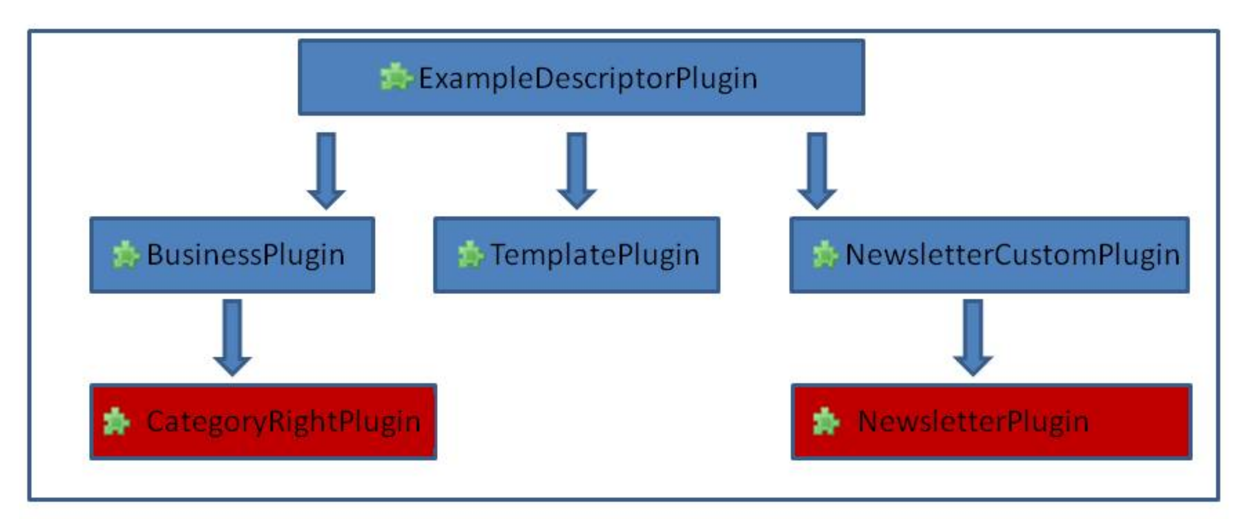
# 7. Utiliser JADE pour mon projet

Cette section détaille comment utiliser JADE pour votre projet en analysant les configurations du build du module et celui de la webapp qui sont fournis par défaut.

## Contexte

La webapp nommée `ExampleWebapp` produite dans la section précédente est constituée comme suit :

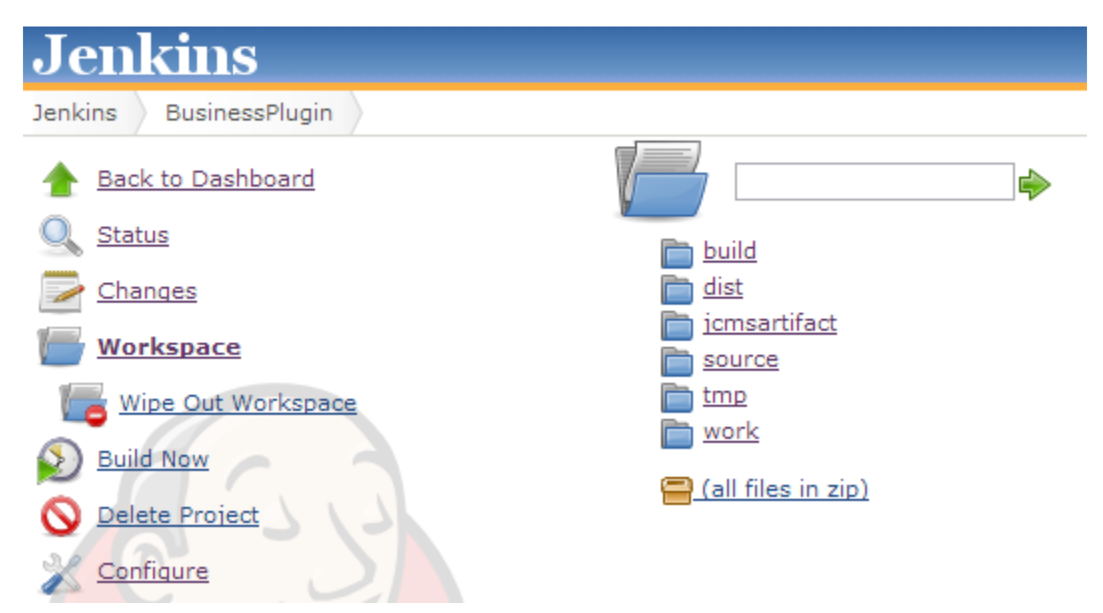
- le module `TemplatePlugin` qui effectue la charte graphique du site.
- le module `BusinessPlugin` gérant les règles de métiers. Ce module dépend de `CategoryRightPlugin`, un module standard fourni par Jalios.
- puis d'autres modules. A titre d'exemple, nous utilisons le module standard de Jalios `NewsletterPlugin` avec un exemple d'extension via `NewsletterCustomPlugin`.
- finalement, un module `ExampleDescriptorPlugin` pour décrire comment construire la webapp




ExampleWebapp

## Comprendre le fonctionnement du build d'un module

Dans cette section, nous détaillerons comment le module `BusinessPlugin` est construit. Dans Jenkins, chaque projet possède un espace de travail. Dans la suite, sauf contre-indication, il faut comprendre que les actions déroulées sont relatives à un espace de travail.




Les étapes de build d'un module sont les suivantes :

1. Téléchargement du code source du module depuis le gestionnaire de version dans le répertoire `source`

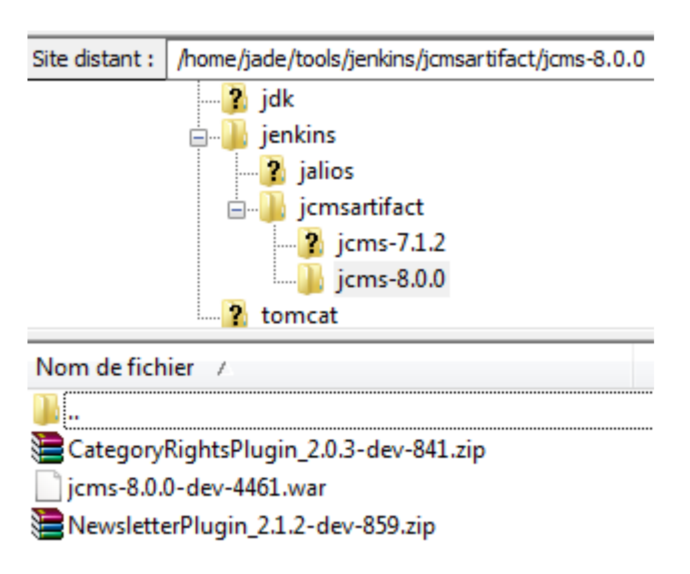
2. Téléchargement du code source de la configuration et du store de référence dans `jcmsartifact`

3. Copie des scripts de build de Jalios dans le répertoire `build`.

   *Note* : ces scripts se trouvent dans le répertoire `/home/jade/tools/jenkins/jalios`

4. Copie des modules de Jalios et la release officielle de JCMS dans le répertoire `jcmsartifact`.

   Important : Ces composants doivent être préalablement déposés via l'utilisateur `jade` dans le répertoire `jcmsartifact` (`/home/jade/tools/jenkins/jcmsartifact`) Plusieurs options sont possibles, le plus de simple est de passer par un client FTP.




5. Création des répertoires nécessaires qui seront utilisés pendant la phase de build. Ils sont nettoyés lors de chaque build.

- `dist` : répertoire stockant les résultats après build. Il contient notamment le résultat des tests unitaires et le build

- `tmp` : répertoire temporaire

- `work` : répertoire de travail pour JCMS

6. Décompression du `.war` JCMS dans le répertoire de travail `work/JCMS`

7. Utilisation des fichiers de références `custom.prop.ref` et `store.xml.ref` en les copiant dans `work/JCMS/WEB-INF/data`

8. Déploiement des modules en cherchant les dépendances si elles existent dans `work/JCMS.` La vérification de la dépendance sera détaillée dans les sections suivantes.

9. Préparation du démarrage d'une instance de Tomcat sur un port libre qui est déterminé automatiquement.

   *Note* : Les erreurs de compilation durant cette étape ne sont pas prises en compte.

10. Démarrage de JCMS via Tomcat qui va utiliser le répertoire `work/JCMS`. Cette étape permet de tester si les classes dans le module sont bien compilées.

    *Note* : Si la compilation échoue, elle va entraîner l'échec du build.

11. Deuxième démarrage de JCMS pour les tests unitaires. Le build prend en charge tous les tests unitaires qui déposés dans le répertoire `unittests` du module.

    Important : Le concept clé de l'intégration continue est de faire des tests en continu pour détecter le plus rapidement des problèmes. Plus la couverture de test est large, moins vous aurez de soucis. [Un article dédié vous aidera à savoir comment faire des tests unitaires dans JCMS.]

```
BusinessPlugin 29 [http://192.168.247.130/svn/repos
  plugins 11
  unittests 31
    com 31
      jade 31
        util 31
          JadeUtilTest.java 32
          TitleControllerTest.java 37
  WEB-INF 29
```

*Note* : Par défaut, si le test unitaire échoue, le build va continuer. Ce comportement est paramétrable en fonction de votre besoin. Pour ce faire :

- Créez un fichier `build.properties` à la racine du projet du module. Dans l'exemple de `BusinessPlugin`, le fichier `build.properties` se trouve au même niveau que les autres répertoires `plugins`, `unittests` et `WEB-INF.`

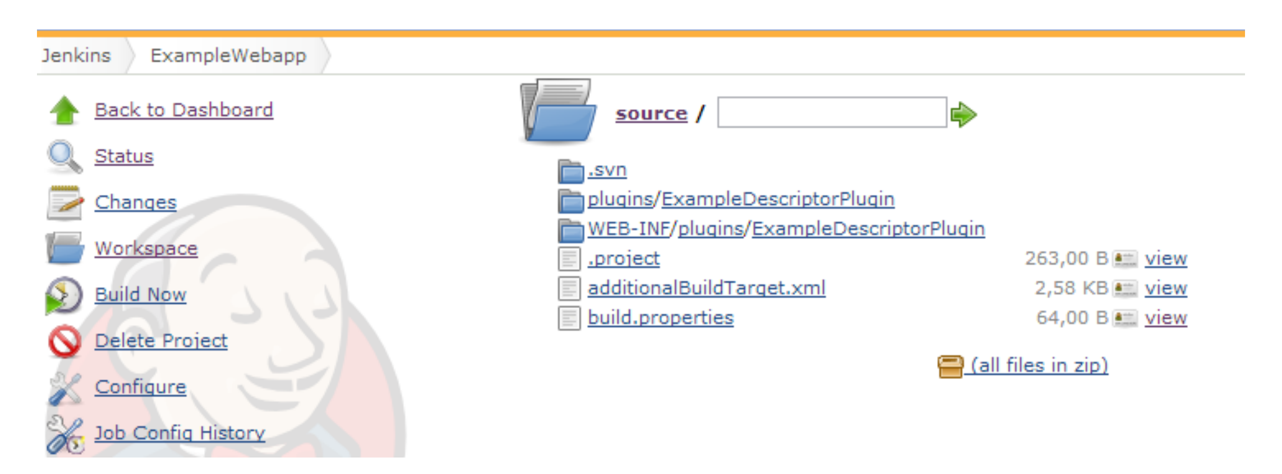
- Ajoutez la ligne `breakBuildOnFailedTest.enabled=true`

A titre de démonstration, nous avons expressément laissé un test unitaire échoué. Dans le log, vous remarquez les lignes suivantes

```
...
[junit] Running com.jade.util.TitleControllerTest
[junit] Tests run: 1, Failures: 1, Errors: 0, Time elapsed: 0,07 sec
[junit] TEST com.jade.util.TitleControllerTest FAILED
[junit] Tests FAILED
[echo] 01/03/2013 17:03:04: END Running junit test for plugin
      ...
```

12. Construction du livrable sous forme de `BusinessPlugin-version-buildnumber.zip`.

## Comprendre le fonctionnement du build d'une webapp




Les étapes de build d'une webapp pour produire un .war sont identiques à celles de build d'un module sauf la fin. C'est le module `ExampleDescriptorPlugin` qui décrit les modules constituant la webapp finale. Nous détaillerons l'utilisation de la description de dépendance ultérieurement.
Une fois que les tests unitaires sont terminés, le build va procéder aux étapes suivantes :

1. Vérification dans `build.properties` s'il s'agit du build d'une webapp via la propriété `build.war.enabled`
2. Si la propriété est positionnée, c'est à dire `build.war.enabled=true`, la construction du war sera déclenchée
3. Création du répertoire temporaire `tmp/PluginWebappWarContent`. Dans la suite, les opérations seront déroulées dans ce répertoire.
4. Décompression du `.war` JCMS
5. Déploiement des modules en cherchant les dépendances
6. Mise à jour du fichier `build.prop` contenant des informations de build
7. Signature de la webapp construite
8. Déclenchement d'un traitement complémentaire via le mécanisme de hook de Jalios `pluginBeforeBuildWebappWarHook`
    1. Dans l'exemple fourni, le traitement complémentaire consiste à utiliser les fichiers `store.xml.ref` et `custom.prop.ref` qui sont considérés comme référence.
    2. Un renommage de l'URID est effectué
    3. Puis un nettoyage du répertoire `WEB-INF/classes/generated`

    *Note* : ce traitement complémentaire est implémenté dans le fichier `additionalBuildTarget.xml` situé dans le module `ExampleDescriptorPlugin`

En résumé, après cette section, vous savez :

- comment un module est construit
- comment intégrer vos tests unitaires pour qu'ils soient exécutés automatiquement durant le build. Vous savez également comment paramétrer le comportement de build par rapport aux résultats des tests unitaires.
- comment une webapp est construite
- comment implémenter un traitement complémentaire lors du build d'un webapp

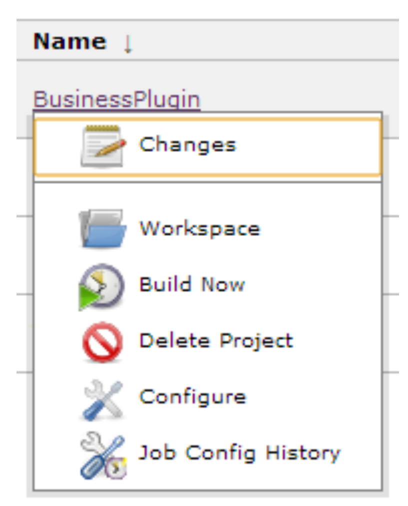
Dans la suite, nous aborderons la configuration de Jenkins. Ces opérations auront pour but de vous montrer comment :

- construire un module ou une webapp lors qu'elle est privée dans la terminologie de JCMS. Pour rappel, une webapp est privée demande systématiquement une authentification
- manipuler les dépendances de JCMS
- mettre en place les builds périodiques qui est un concept important de l'intégration continue
- manipuler le fichier build.properties exploité par les scripts JADE

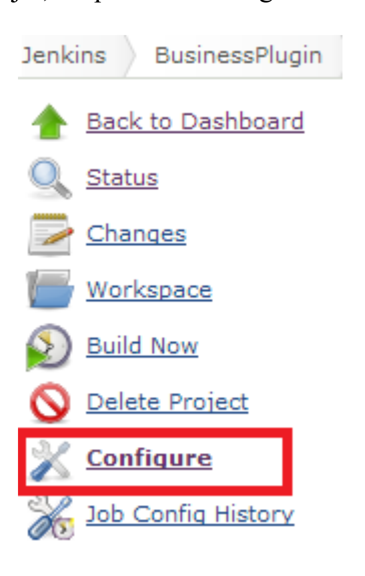
## Comprendre la configuration d'un projet de Jenkins

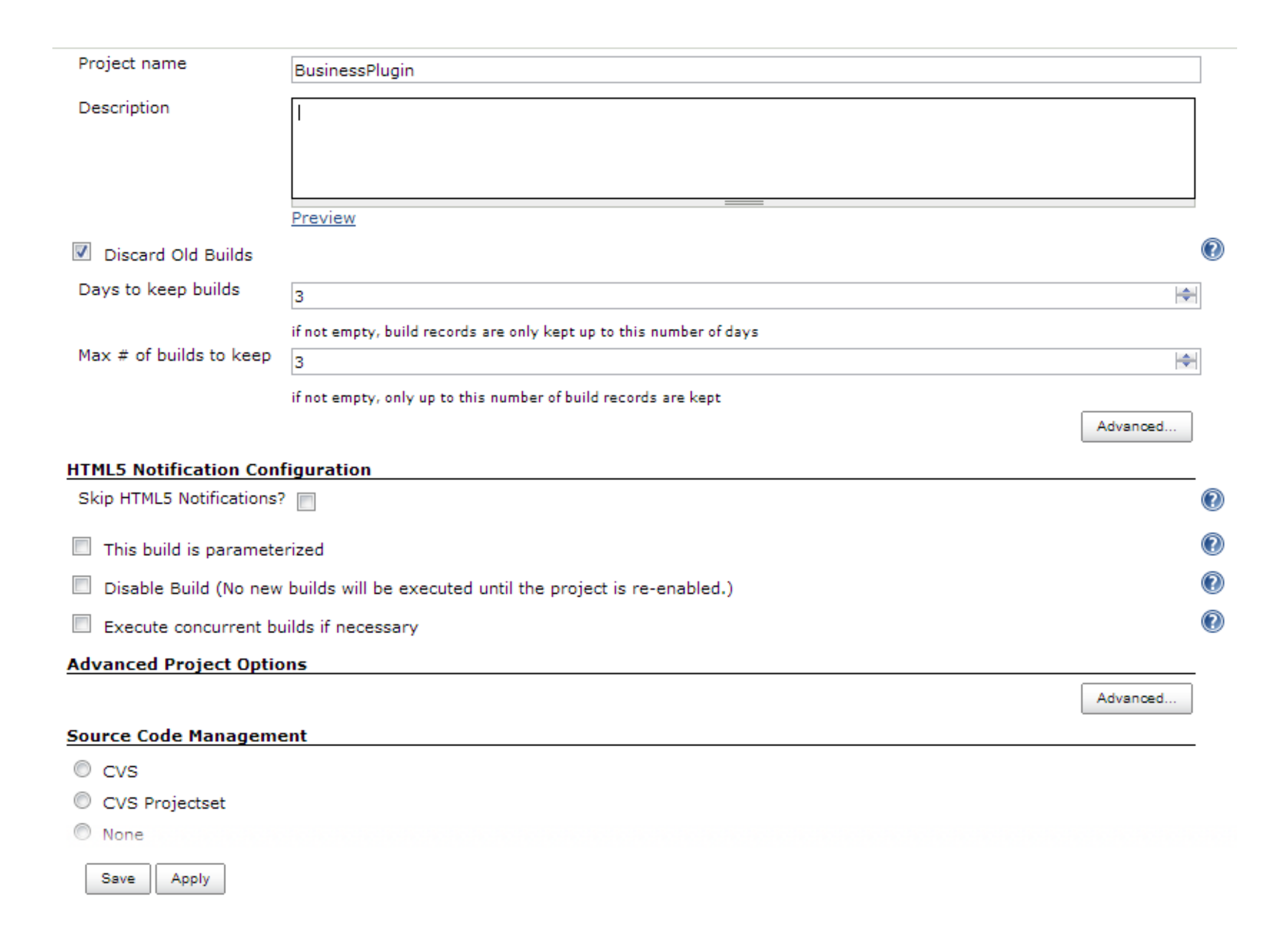
Pour configurer un projet, plusieurs options sont possibles :

- depuis le tableau de bord, survolez sur le nom du projet pour avoir le menu contextuel. Cliquez sur Configure




- ou bien depuis la vue du projet, cliquez sur Configure






## Nom du projet

Respectez la nomenclature : NamePlugin

*Note : Ne pas mettre d'accent ni d'espace dans le nom du projet.*

## Période de garde de build

Configurez la période de garde des anciennes builds (soit en nombre de jours, et/ou en nombre de builds). Plus vous augmentez la période, plus Jenkins prend de l'espace pour stocker les builds.

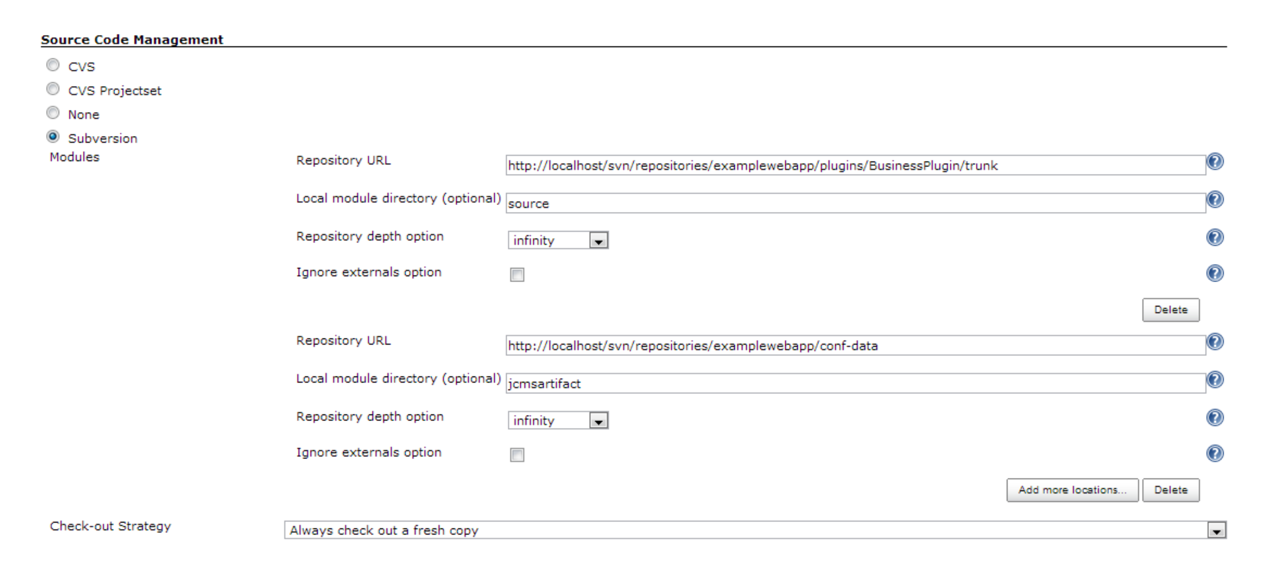
Faites une estimation et contactez votre exploitation pour déterminer le seuil raisonnable. Une période entre 7 et 14 jours est raisonnable.

## Gestionnaire de version

Durant le build, Jenkins est capable d'aller récupérer le code source des gestionnaires de version. Dans le cadre de JADE, nous avons intégré un serveur SVN pour la démonstration.

**Deux points importants** à noter :

- le code source doit être téléchargé dans l'espace de travail dans le répertoire `source`
- la configuration et le store de référence qui se trouve sur SVN à `conf-data` doivent être téléchargés dans le répertoire `jcmsartifact`



## Configuration de déclenchement

**Build Triggers**

- [ ] Build after other projects are built
- [ ] Trigger builds remotely (e.g., from scripts)
- [x] Build periodically

Schedule

- [x] Poll SCM

Configure Jenkins to poll changes in SCM.

Note that this is going to be an expensive operation for CVS, as every polling requires Jenkins to scan the entire workspace and verif
trigger to avoid this overhead, as described in this document

Schedule

Ignore post-commit hooks

Les déclenchements sont au format crontab. Il est conseillé d'effectuer deux déclenchements :

1. un déclenchement périodique via l'option `Build Triggers > Build periodically`

2. un déclenchement après scrutation de SVN via l'option `Poll SCM`

Ils permettent de vérifier que les dernières modifications effectuées dans JCMS ne rentrent pas en conflit avec ce module.

## Configuration des actions de build

**Build Triggers**

- [ ] Build after other projects are built
- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build periodically
- [ ] Poll SCM

**Build**

**Execute shell**

Command

```
echo "Cleaning and updating Jalios scripts"
rm -rf build
mkdir build
cp -r $JALIOS_SCRIPTS/* build/


echo "Updating Jalios Artifact"
cp -r $JCMS_ARTIFACT/jcms-8.0.0/* jcmsartifact/
```

See the list of available environment variables

**Invoke Ant**

| | |
|---|---|
| Ant Version | Ant |
| Targets | buildPlugin |
| Build File | ../build/buildPlugin.xml |

Il y a deux actions de build :

- Les scripts shell qui nettoyent et copient les scripts de Jalios et les composants de JCMS.
- Le script Ant qui se charge de l'intelligence de construction

### Configuration des actions post build

**Post-build Actions**

**Archive the artifacts**

Files to archive **/dist/*.zip

**Publish JUnit test result report**

Test report XMLs dist/testReport/*.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such a

☑ Retain long standard output/error

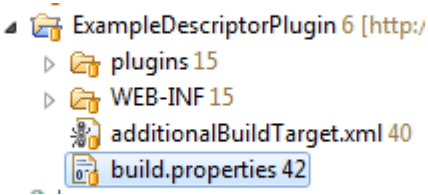
Additional test report features ☑ Publish test attachments

Les actions après avoir construit un module ou webapp sont les suivantes :

- Archiver les résultats pour qu'ils puissent être téléchargés plus tard
- Publier les rapports de résultats des tests JUnit
  - L'option `Retain long standard output/error` permet de voir plus de détails au niveau de log de test JUnit. Attention, cette option est consommatrice et ralentit le build. A utiliser seulement dans les cas particuliers. Il serait aussi intéressant d'archiver le fichier de logs du tomcat (work/catalina/logs/catalina.out) pour des analyses ultérieures.

## Configuration via les fichiers build.properties et additionalBuildTarget.xml

En dehors de la configuration de build d'un job, vous pouvez paramétrer le comportement de build via deux autres fichiers qui sont à la racine de l'arborescence du module.




### build.properties

Ce fichier permet de surcharger les propriétés définies par défaut. Les propriétés souvent surchargées sont les suivantes :

| Nom de la propriété | Fonctionnement |
|---|---|
| `isPrivate` | builder une webapp privée |
| `breakBuildOnFailedTest.enabled` | arrêter le build lors qu'un test unitaire échoue |
| `webappName` | Nommer le nom de la webapp |
| `build.war.enabled` | Builder un .war |

### additionalBuildTarget.xml

C'est le mécanisme de hook de Jalios. Ce fichier est destiné à personnaliser le comportement de construction du .war.

```
<project basedir=".">
  <target name="pluginBeforeBuildWebappWarHook">
...
```

## Configuration de dépendances

La dépendance est gérée à deux niveaux : descripteur de la webapp et configuration du job.

### Niveau du descripteur de la webapp.

Dans `ExampleDescriptorPlugin`, le plugin.xml contient la déclaration suivante pour indiquer les modules dont la webapp a besoin.

```
<dependencies>
    <dependency name="BusinessPlugin" />
    <dependency name="TemplatePlugin" />
    <dependency name="NewsletterCustomPlugin" />
</dependencies>
```

Concernant les modules venant de Jalios, il existe une seule version officielle correspondant à une JCMS c'est pourquoi il suffit de déposer ces modules dans le répertoire `jcmsartifact`.
Par contre, concernant les modules développés spécifiquement, nous avons besoin de spécifier les versions à récupérer. Cela va se passer via la configuration du job de la webapp.

### Configuration de dépendance via un job

Pour construire une webapp, le script Ant va chercher dans le répertoire `jcmsartifact` tous les modules dont le descripteur du module a besoin.
Concernant les modules développés spécifiquement, ils sont construits dans Jenkins. Pour les copier dans le répertoire `jcmsartifact`, il faut passer par un mécanisme de copie dédié. Le comportement de la copie permet de mentionner :

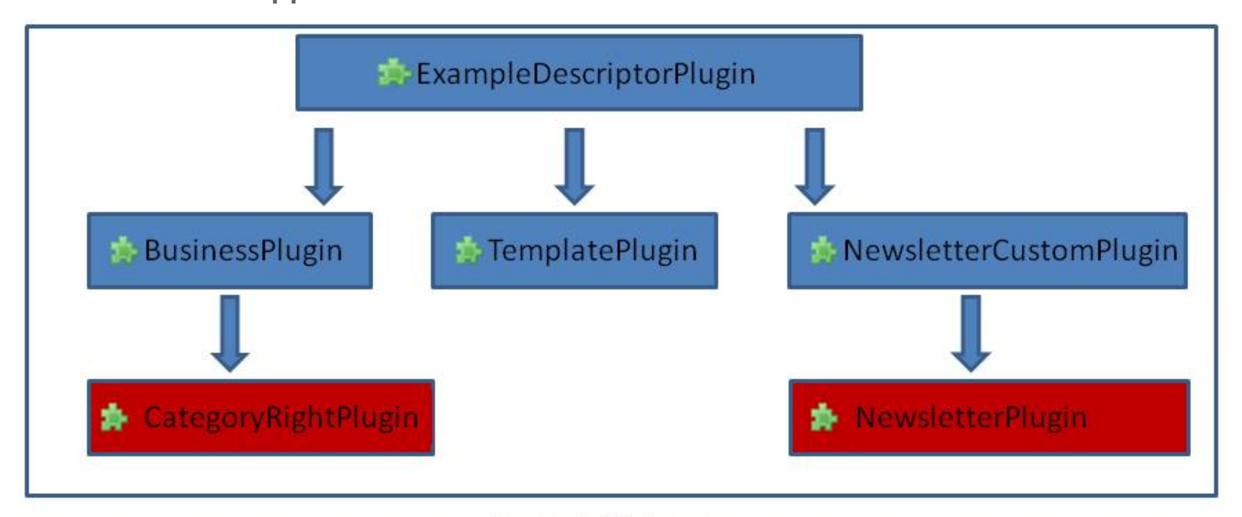
- le projet à copier
- la version du build à copier. Jenkins offre plusieurs notions de version (dernier build en succès, build marqué à garder) permettant de construire dynamiquement la webapp

**Copy artifacts from another project**
Project name: TemplatePlugin
Which build: Latest successful build
☐ Stable build only
Artifacts to copy:
Target directory: jcmsartifact
☑ Flatten directories ☐ Optional

**Copy artifacts from another project**
Project name: BusinessPlugin
Which build: Latest saved build (marked "keep forever")
Artifacts to copy:
Target directory: jcmsartifact
☑ Flatten directories ☐ Optional

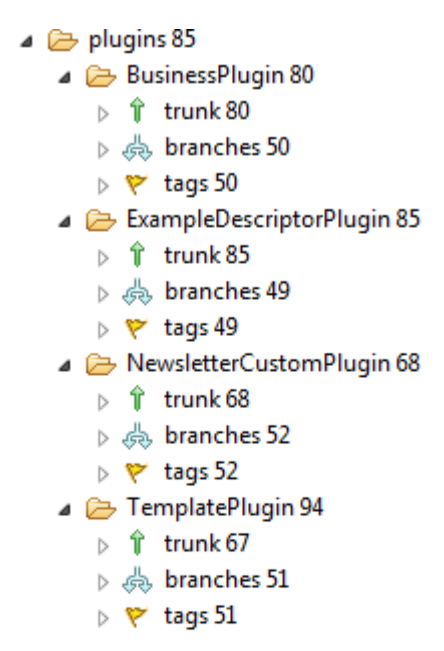
## Passer mon projet actuel dans JADE

### Rendre le développement modulaire




Cette étape consiste à déterminer les modules :

- fournis par Jalios. Dans le cas de la webapp `ExampleWebapp`, ce sont les modules `CategoryRightPlugin` et `NewsletterPlugin`
- développés spécifiquement dans votre contexte. Dans l'exemple, ce sont les modules : `ExampleDescriptorPlugin`, `BusinessPlugin`, `TemplatePlugin` et `NewsletterCustomPlugin`. Ces modules doivent être dans votre gestionnaire de version.



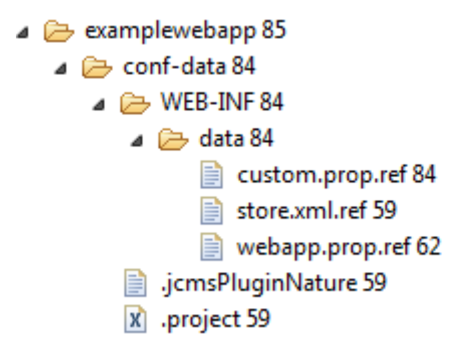

*Note* : Pour plus de détails de mise en place des modules, consultez JCMS - Développement orienté module avec Eclipse et SVN

## Identifier les configurations et le store.xml

Cette étape consiste à mettre dans le gestionnaire de version les fichiers propriétés et le `store.xml` sous forme de fichiers de références.

**Attention :** Il faut les merger préalablement avec celles de la nouvelle version de JCMS.

*Note* : Pour plus de détails de mise en place des, consultez JCMS - Développement orienté module avec Eclipse et SVN



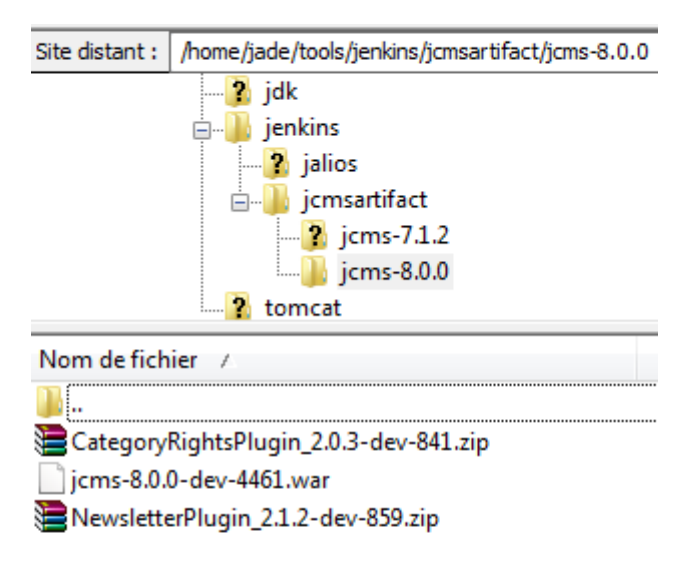

## Monter le répertoire jcmsartifact

C'est le répertoire où se trouve les modules de Jalios et la release officielle de JCMS qui est situé à `/home/jade/tools/jenkins/jcmsartifact.`

Procédez aux étapes suivantes pour peupler ce répertoire :

1. Connectez en tant que l'utilisateur `jade` sur le serveur JADE via un client FTP. Attention, c'est l'utilisateur `jade` du système d'exploitation.

2. Allez dans le répertoire `jcmsartifact` situé à `/home/jade/tools/jenkins/jcmsartifact`




3. Créez un répertoire par version JCMS en dessous de `jcmsartifact`

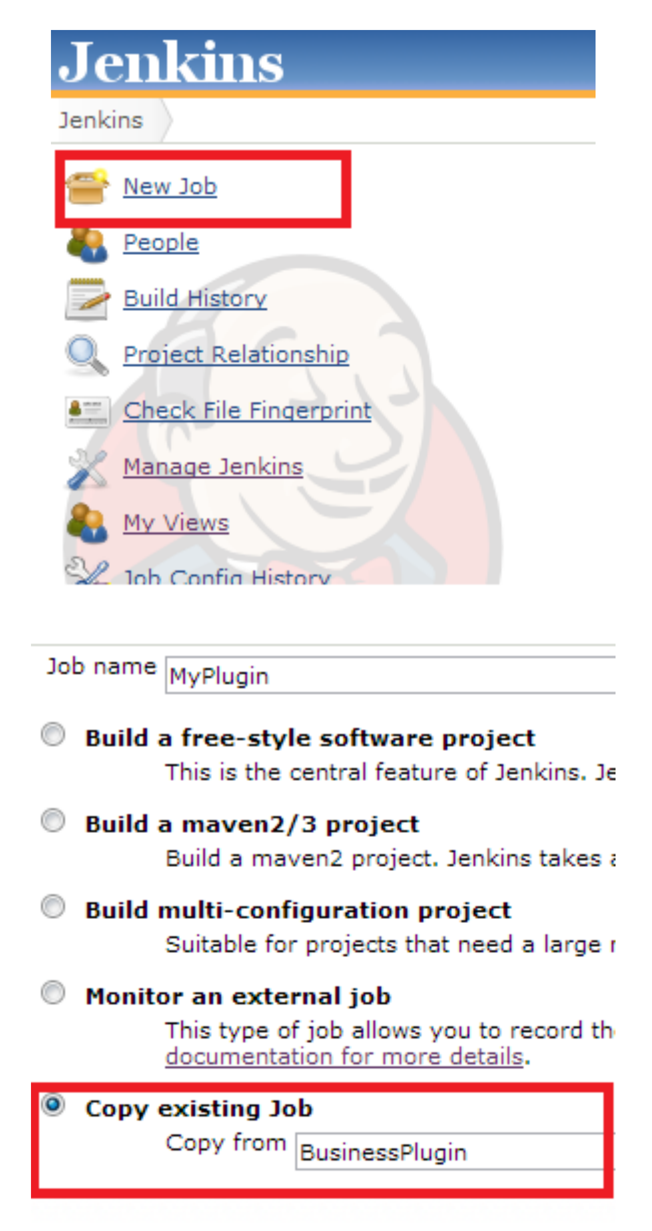
4. Déposez la release officielle de JCMS et les modules associées qui sont nécessaires pour le développement du projet

## Créer le job / la configuration de build d'un module dans Jenkins

Le terme `job` désigne la configuration d'un projet sous Jenkins. C'est un ensemble de règles (copier un fichier, faire un update de SVN, etc...) qui seront appliquées lors d'un build.

Procédez aux étapes suivantes pour créer un job de build d'un module :

1. Créez un nouveau job en copiant le job `BusinessPlugin` existant fourni dans l'exemple

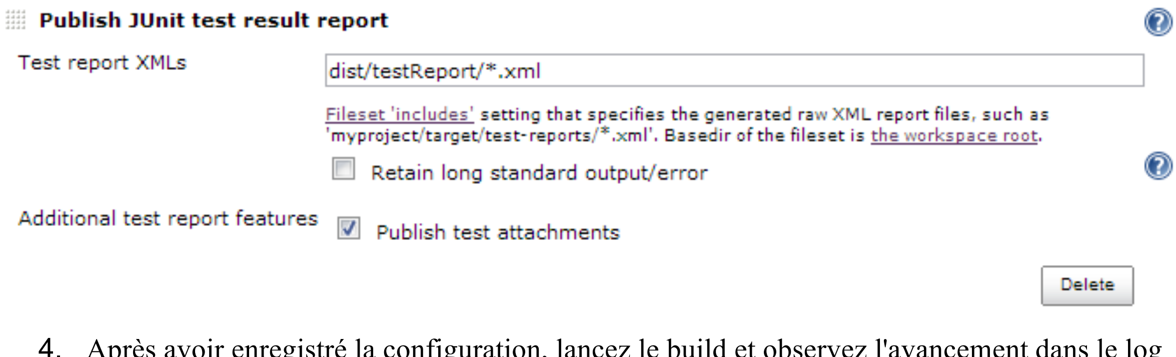



2. Adaptez la section `Source Code Management` pour pointer vers le bon chemin selon votre configuration de gestionnaire de version. Utiliser l'option `Enter credential` pour pouvoir authentifier sur votre gestionnaire de version.

https://your.svn.com/repos/jcms/plugins/YourPlugin

**Unable to access https://your.svn.com/repos/jcms/plugins/YourPlugin : svn: E175002: OPTIONS /repos/jcms/plugins/YourPlugin failed (show details)**
**(Maybe you need to enter credential?)**

3. Si le module n'a pas de test unitaire, il faut supprimer la section `Publish JUnit test result report` > `Delete` > `Save`.




4. Après avoir enregistré la configuration, lancez le build et observez l'avancement dans le log

5. Il est possible d'être notifié. Pour ce faire, ajoutez l'action post-build `E-mail Notification`.

E-mail Notification

Recipients mail@company.com

Whitespace-separated list of recipient addresses. May reference build parameters like $PARAM. E-mail

Send e-mail for every unstable build

Send separate e-mails to individuals who broke the build

Add post-build action

## Créer un job de build de la webapp dans Jenkins

Procédez aux étapes suivantes pour créer un job de build d'une webapp:

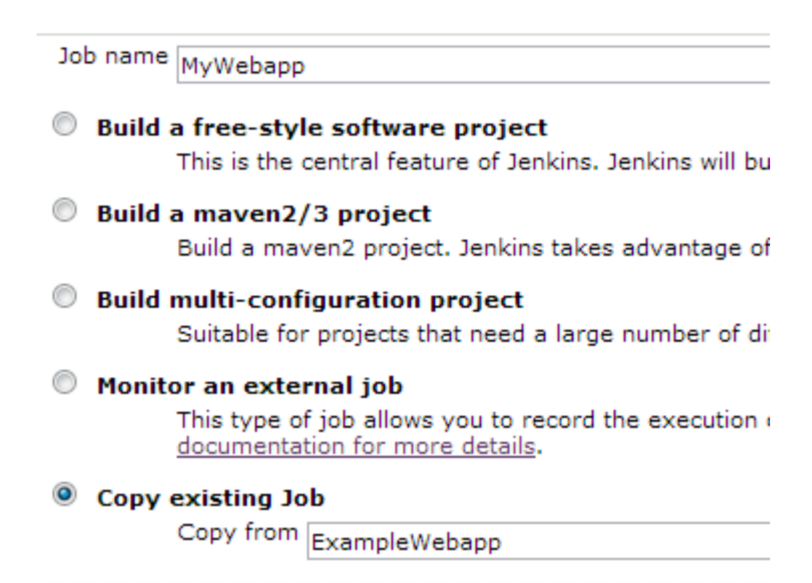
1. Dans votre module descripteur de la webapp, supposons que son nom est `MyDescriptorPlugin`, copiez les deux fichiers de configurations `build.properties` et `additionalBuildTarget.xml` provenant de `ExampleDescriptorPlugin`

2. Editez `build.properties` et modifiez le nom de la webapp à produire via la propriété `webappName`

3. Si la webapp est privée, positionnez la propriété `isPrivate=true`

4. Committez les modifications dans SVN

5. Créez un nouveau job avec un nom selon votre choix, par exemple `MyWebapp`, en copiant `ExampleWebapp`




6. A la section `Source Code Management`, pointez vers l'URL du module DescriptorPlugin sur SVN. Pour rappel, ce module permet d'indiquer comment la webapp sera construite en se basant sur la notion de dépendance de JCMS.

Repository URL http://yoursvn/svn/repositories/plugins/YourDescriptorPlugin/trunk

Unable to access http://yoursvn/svn/repositories/plugins/YourDescriptorPlugin/trunk : svn /svn/repositories/plugins/YourDescriptorPlugin/trunk failed (show details)
(Maybe you need to enter credential?)

7. A la section `Copy artifacts from another project`, définissez le comportement de copie de dépendance pour les développements spécifiques et cochez la case `Flatten directories`. Dans notre exemple, la définition est la suivante :

   - Copie du dernier build réussi du module `TemplatePlugin` dans le répertoire `jcmsartifact`

- Copie du dernier build stable qui est marqué comme à garder pour toujours du module `BusinessPlugin` dans le répertoire `jcmsartifact`.

**Copy artifacts from another project**

Project name: TemplatePlugin

Which build: Latest successful build

☐ Stable build only

Artifacts to copy:

Target directory: jcmsartifact

☑ Flatten directories ☐ Optional

**Copy artifacts from another project**

Project name: BusinessPlugin

Which build: Latest saved build (marked "keep forever")

Artifacts to copy:

Target directory: jcmsartifact

☑ Flatten directories ☐ Optional

8. Supprimez éventuellement les autres définitions de copie lors de la duplication du job

**Copy artifacts from another project**

Project name: BusinessPlugin

Which build: Latest saved build (marked "keep forever")

Artifacts to copy:

Target directory: jcmsartifact

☑ Flatten directories ☐ Optional

Delete

# 8. Déploiement en continue

Dans cette version de JADE, nous avons inclus deux nouveaux modules :

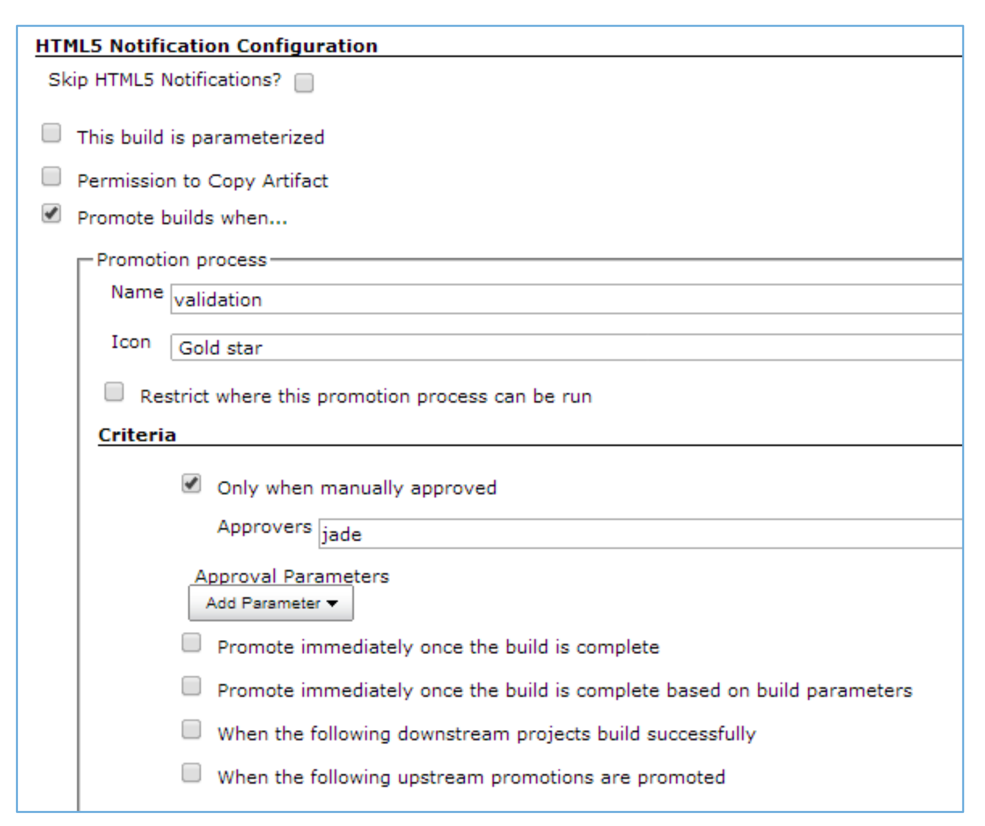
- Promoted Builds Plugin : permet de distinguer un bon build et un mauvais en introduisant la notion de « promotion ». Dans la suite, nous allons utiliser cette notion pour déclencher un traitement particulier
- Publish Over SSH Plugin : permet de copier un fichier sur un serveur ssh à distance

Dans la suite, en combinant ces deux modules, nous pouvons parvenir au scénario suivant :

- Une version JCMS a besoin d'être déployée sur la recette et un responsable doit l'approuver le build
- Suite à cette action, le .war sera envoyé par SSH sur le serveur de recette
- Ensuite, un script sera appelé automatiquement depuis le serveur de recette. Ce script est normalement un « upgrade.sh » qui redéployer le war et redémarrer le serveur.

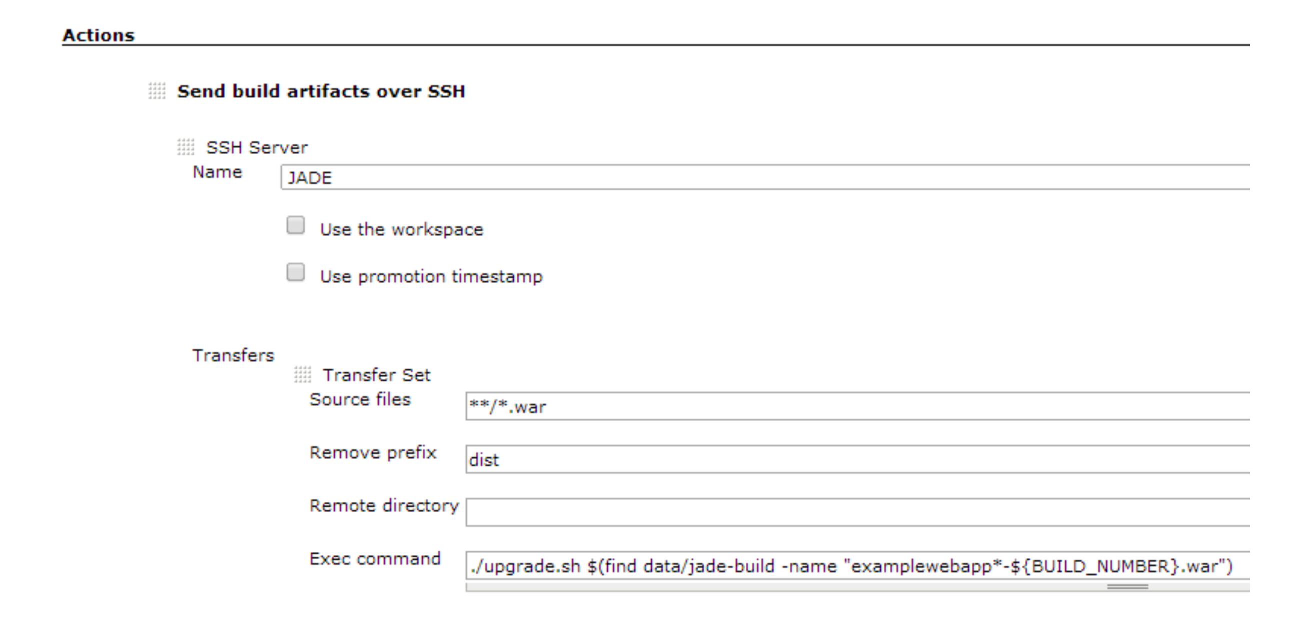
Pour ce faire :

- Editer le projet « ExampeWebapp »
- Repérer la section « HTML5 Notification Configuration » et cocher « Promote builds when… »
- Au niveau de « Criteria », cocher « Only when manually approved » et renseigner le champ « Approvers ». Dans notre exemple, c'est « jade »




- Au niveau de « Actions », configurer le serveur. Ce sont des informations à renseigner au niveau de l'administration de Jenkins.

- Enfin, renseigner le champ « Exec command » pour lancer le script désiré. Dans notre exemple, nous allons lancer un script « upgrade.sh » sur le war qui vient d'être déposé sur le serveur.

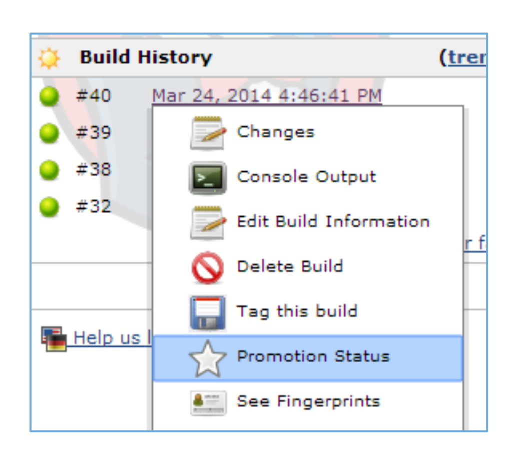


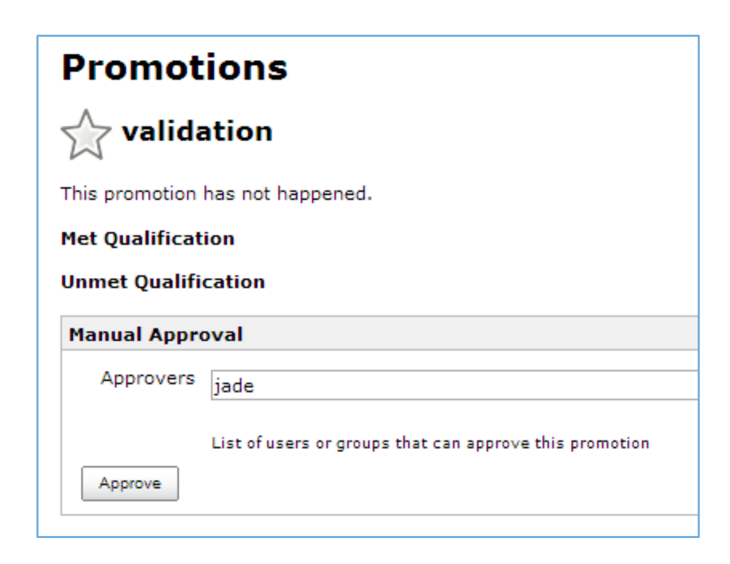

- Attention, le script lancé à distance doit finir et redonner la main à Jenkins. Donc il faut se méfier des commandes comme « tail » qui consiste à afficher le log. Une astuce pour visualiser les logs depuis Jenkins :
  - Dans le script « upgrade.sh », inclure un script « logs.sh »
  - Contenu de « logs.sh »

```
tail --pid=$$ -f logs/catalina.out | { sed "/INFO: Server startup/ q" && kill $$ ;}
```

  - L'enchaînement de deux commandes permet d'afficher les logs jusqu'à ce que la chaîne « INFO : Server startup » apparaît.

Une fois que la configuration est mise en place, revenir sur le build que nous souhaitons promouvoir. Attention, l'option « Promotion Status » est seulement disponible pour les builds lancés après la mise en place de « Promoted Build Plugin »






Pour résumer, nous avons mis en place un scénario où il faut une validation pour déployer sur le serveur de recette. Pour le déploiement en continu, il suffit d'enlever la notion de validation.
Désormais, nous sommes donc capable de mesurer le temps pour qu'un commit de code parte en recette.

# 9. Procédures d'exploitation

## Les administrateurs

Il y a deux comptes `jade` :

- L'un pour connecter dans Jenkins
- L'autre pour connecter sur la VM

Le mot de passe par défaut pour ces comptes est `jade`. Il est indispensable de regénérer un mot de passe sécurisé lors de la première installation et utilisation.

## JADE

Le démarrage et l'arrêt de JADE consistent simplement à démarrer et arrêter la VM.

### Démarrage

Via vSphere client ou une console d'administration shell VMWare (vSphere CLI). Procédez aux étapes suivantes :

1. Cliquez-droit sur la VM puis en sélectionnant `Power On` ou en cliquant sur l'icône 
2. Via le CLI : « `vim-cmd vmsvc/power.on <vmid>` »

*Note :* <vmid> est l'id de machine virtuelle, pouvant être obtenu via la commande « `vim-cmd vmsvc/getallvms | grep <vm_name>` » où <vm_name> correspond au nom de la VM

## Redémarrage

1. Connectez-vous au shell de la machine avec le compte d'administration `jade` via SSH ou la console VMware de l'ESX

2. Lancez la commande suivante

   ```
   $> sudo shutdown -r now
   ```

## Arrêt

Pour arrêter la machine, lancez la commande :

```
$> sudo shutdown -h now
```

## Jenkins

### Démarrage de Jenkins

Pour démarrer Jenkins :
```
$> /home/jade/tools/jenkins/start_jenkins.sh
```

### Arrêt de Jenkins

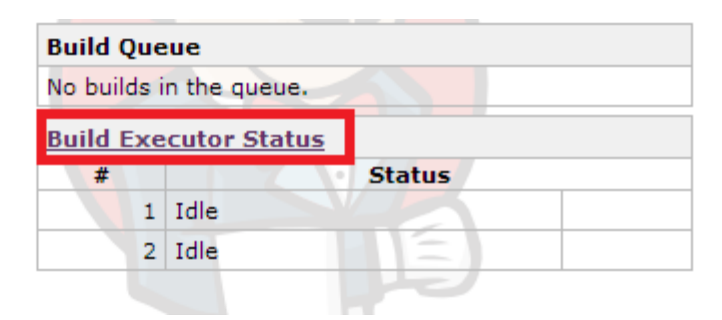
Procédez à l'arrêt de Jenkins en suivant les étapes suivantes :

1. Connectez-vous dans Jenkins en tant qu'administrateur
2. Allez dans `Manage Jenkins`
3. Cliquez sur `Prepare for Shutdown.` Cette action interdit les nouveaux builds de démarrer. Eventuellement, il faut attendre que les builds en cours se terminent avant de procéder à l'arrêt complet de Jenkins.
4. Arrêtez Jenkins en lancant la commande suivante :

   ```
   $> /home/jade/tools/jenkins/stop_jenkins.sh
   ```

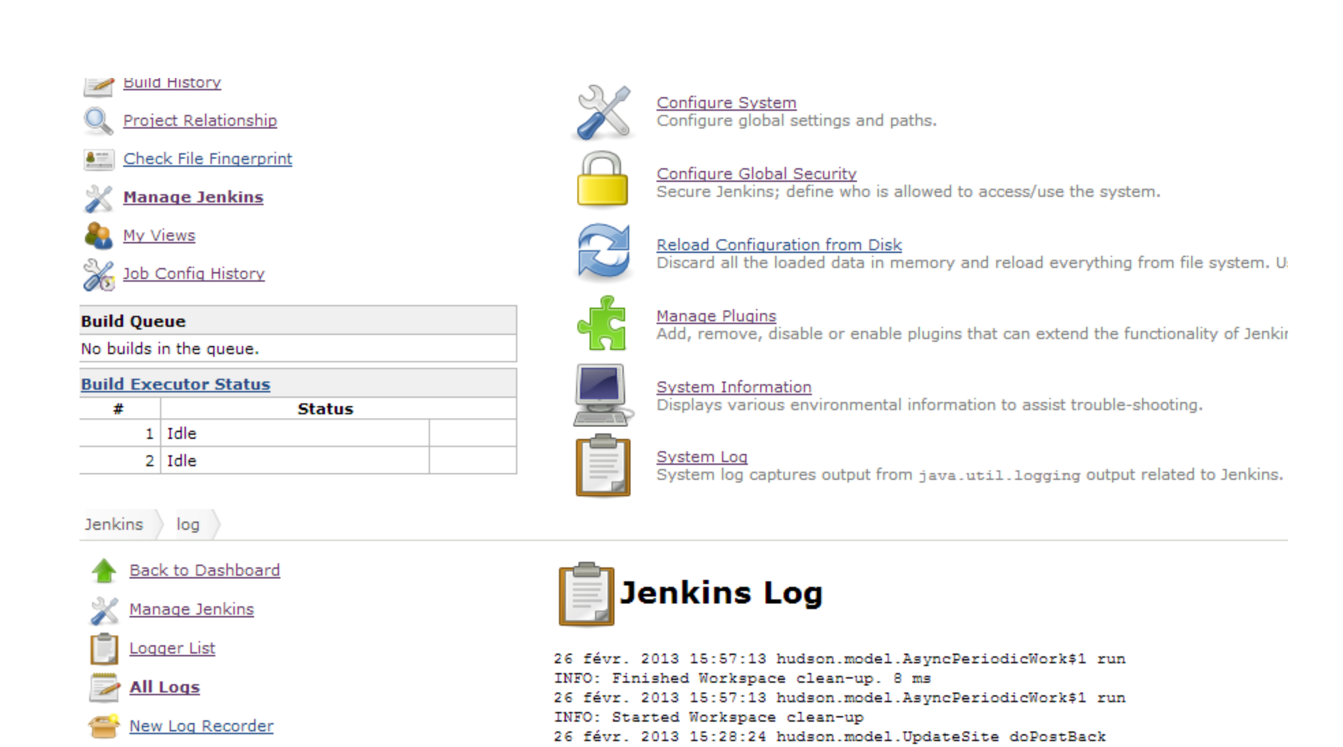
### Paramétrage de l'allocation mémoire

En soi, Jenkins n'est pas consommatrice en mémoire. Ce sont les activitiés de build qui en nécessitent. Plus il y a de buid en parallèle, plus il faut de la mémoire. Il faut assurer que le serveur possède suffisamment de RAM pour supporter la charge des builds simultanés.
Dans cette version de JADE, chaque build possède par défaut 512Mo lors de son exécution. Ce paramètre est réglable en utilisant la propriété `catalinaOpts` dans le fichier `build.properties`.
Le nombre d'exécuteur de build est à 2 par défaut. En fonction de la mémoire disponible sur JADE, vous pouvez ajuster ce paramètre pour avoir plus de build en parallèle.




### Journal des évènements

Jenkins permet de visualiser en temps réel ses logs pour diagnostiquer les dysfonctionnements. Pour ce faire, connectez en tant qu'administrateur et allez dans `Manage Jenkins > System Log.`
Vous pouvez vous inscrire aux fils RSS pour surveiller les différents niveaux de logs (ERROR, WARNING).




*Note :* Sur JADE, à l'installation de Jenkins, les logs se trouvent dans le répertoire suivant : `/var/log/jenkins/jenkins.log`

Les autres logs consultables lors de l'échec d'un build :

- Sortie Console du build accessible dans la colonne de gauche au niveau du build concerné
- Logs de JCMS : allez dans l'espace de travail du projet puis consultez le fichier `work/JCMS/WEB-INF/data/logs/jcms.log`
- Logs de tomcat : allez dans l'espace de travail du projet puis consultez le fichier `work/catalina/logs/catalina.out`

## ESX

La sauvegarde et la restauration des VMs peut se faire soit directement avec VMWare, soit avec un outil tiers, comme VMware Data Protection, Quest vRanger Pro, Veeam Backup, Acronis vmProtect, etc… Nous allons aborder ici la sauvegarde et la restauration avec les outils de base de VMWare.

### Sauvegarde de la VM

Sans outil particulier, il existe 3 solutions :

1. Arrêter la VM et la cloner :
   - Exécuter via SSH ou la console VMware « sudo shutdown –h now » ;
   - Aller sur la VM dans vSphere Client, et faire un clic-droit + « Cloner » ;
   - Choisir le nom et l'emplacement du clone puis cliquer sur « Suivant » ;
   - Choisir l'ESX sur lequel le clone doit être copié ;
   - Choisir le format de disque et la banque de stockage, puis cliquer sur « Suivant » ;
   - Choisir éventuellement les options de personnalisation, puis cliquer sur « Suivant » ;
   - Vérifier le résumé des paramètres et cliquer sur « Terminer ».
2. Arrêter la VM et l'exporter comme modèle OVF :
   - Aller sur la VM dans vSphere Client et choisir « Exporter / Exporter modèle OVF… »
   - Choisir le nom du fichier, l'emplacement où il doit être sauvegardé et le format de sauvegarde (OVF = dossier de fichiers, OVA = un fichier unique). **Attention** : l'espace disponible dans le dossier de destination doit être capable d'accueillir les fichiers de la VM, suivant le mode de provisionnement que vous avez choisi ;
   - Remplir éventuellement une description et cliquer sur « OK » pour que l'export démarre.
3. Installer sur l'ESX le script ghettoVCB (voir ici pour l'installation et la documentation : http://communities.vmware.com/docs/DOC-8760)

**Attention** : ces solutions ne permettent pas d'accéder au système de fichiers de la VM, et ne permettent donc pas la restauration d'un ou plusieurs fichiers spécifiques. On sauvegarde et on restaure obligatoirement toute la VM.

## Restauration de la VM

Suivant la méthode de sauvegarde choisie ci-dessus :

1. S'il s'agit d'un clone :

   - Arrêter ou supprimer la VM d'origine ;
   - Paramétrer le clone pour que les informations liées au réseau soient identiques (adresse MAC, connexion réseau) ;
   - Cliquer sur démarrer. Au bout de quelques secondes, vous aurez la VM telle qu'elle était au moment de la sauvegarde.

2. S'il s'agit d'un modèle OVF/OVA :
   - Arrêter ou supprimer la VM d'origine
   - Choisir « Fichier / Déployer modèle OVF » ;
   - Sélectionner ensuite le fichier jade.ova enregistré au point 1 et cliquer sur « Suivant » ;
   - Cliquer une nouvelle fois sur « Suivant » ;
   - Choisir ensuite un nom et un emplacement pour la VM, puis cliquer sur « Suivant » ;
   - Choisir l'hôte sur lequel la machine doit être déployée et cliquer sur « Suivant » ;
   - Choisir le type de provisionnement pour le stockage de la VM, suivant les règles que vous utilisez pour vos VMs.
   - Choisir le réseau sur lequel la VM doit être connectée, puis cliquer sur « Suivant » ;
   - Vérifier le résumé d'installation pour valider tous les choix effectués, puis cliquer sur « Terminer ». Vous pouvez activer « Mettre sous tension après le déploiement » si vous désirez que la VM démarre immédiatement après son installation sur l'ESX.
3. S'il s'agit de ghettoVCB : lisez la documentation correspondante.

## SVN

**Important** : La sauvegarde doit être stockée en dehors de la VM JADE

### Sauvegarde de SVN

JADE intègre par défaut SVN pour faciliter la démonstration. Il est recommandé d'avoir un serveur en dehors de JADE et c'est le cas de la plupart des organismes.

Avec SVN, il faudrait passer par une procédure de dump. La commande `svnadmin` qui permet à l'aide de l'option `dump` de copier dans un simple fichier le contenu de la base de donnée.

Procédez aux étapes suivantes pour sauvegarder :

1. Allez dans `/home/jade/data/svn`
2. Lancez la commande suivante

```
$> sudo svnadmin -q dump repositories > /backup/jaliosrepo.svndump
```

### Restauration de SVN

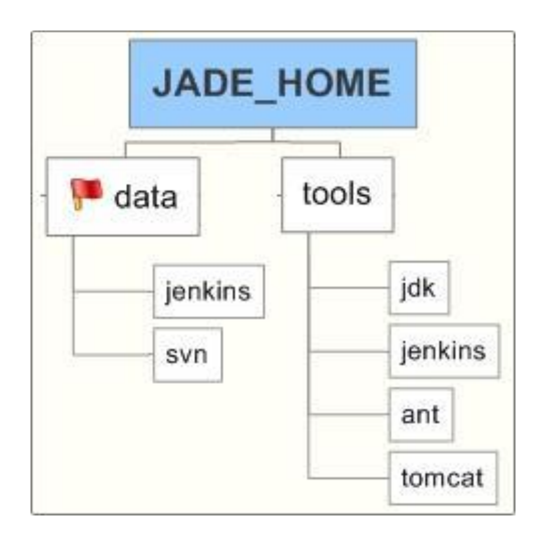
Procédez aux étapes suivantes pour sauvegarder :

1. Allez dans `/home/jade/data/svn`
2. Lancez la commande suivante

```
$> sudo svnadmin load repositories < /backup/jaliosrepo.svndump
```

3. Vérifiez les droits des sous répertoires

## Jenkins




### Sauvegarde de Jenkins

Les données et la configuation de jenkins se trouve dans le répertoire `/home/jade/data/jenkins`. Les configurations sont stockées sous forme d'ensemble de fichiers .xml.

Il suffit de sauvegarder ce répertoire avec votre politique de sauvegarde.

**Important** : La sauvegarde doit être stockée en dehors de la VM JADE

*Note* : La sauvegarde peut être effectuée pendant que Jenkins est en marche. Il n'est pas nécessaire d'arrêter le serveur.

### Restauration de Jenkins

Procédez aux étapes suivantes pour restaurer Jenkins :

1. Connectez-vous avec l'utilisateur `jade` dans Jenkins
2. Allez dans Manage Jenkins puis cliquez sur `Prepare for Shutdown`
3. Attendez que les builds s'arrêtent
4. Allez au répertoire /home/jade/tools/jenkins et arrêtez Jenkins via le script `stop_jenkins.sh`
5. Remettez la sauvegarde dans le répertoire `/home/jade/data/jenkins`. Attention à respecter la structure d'arborescence d'origine de Jenkins.

```
Fingerprint cleanup.log
fingerprints/
hudson.maven.MavenModuleSet.xml
hudson.model.UpdateCenter.xml
hudson.scm.CVSSCM.xml
hudson.scm.SubversionMailAddressResolverImp
hudson.scm.SubversionSCM.xml
hudson.tasks.Ant.xml
hudson.tasks.Mailer.xml
hudson.tasks.Maven.xml
hudson.tasks.Shell.xml
hudson.triggers.SCMTrigger.xml
identity.key
jenkins.model.JenkinsLocationConfiguration.
jenkins.security.RekeySecretAdminMonitor/
jobs/
nodeMonitors.xml
org.jenkins.ci.plugins.html5_notifier.Globa
plugins/
queue.xml.bak
secret.key
secrets/
updates/
userContent/
users/
Workspace clean-up.log
```

6. Mettez les droits pour l'utilisateur jenkins.

```
$> sudo chown -R jenkins:jenkins /home/jade/data/jenkins
```

# 10. Mise à jour de JADE

La mise à jour de JADE consiste essentiellement à mettre à jour Jenkins.

## Mise à jour de Jenkins

Procédez aux étapes suivantes pour mettre à jour Jenkins :

- Téléchargez sur le site de support de Jalios :
  - le script de mise à jour `upgradeJADE.sh`
  - l'archive de mise à jour de JADE sous forme `upgrade-jade-version-to-version.zip`. Par exemple `upgrade-jade-1.0-to-1.1.zip`
- Consultez la section de Jenkins dans l'article dédié à la mise jour de JADE pour plus de détails

## Mise à jour de JDK

Il s'agit de la JDK utilisée les projets JCMS. La JDK est installée dans le répertoire `/home/jade/tools/`

Procédez aux étapes suivantes pour mettre à jour JDK :

- Consultez le manuel d'exploitation sur le site de support Jalios pour connaître la version JDK à télécharger sur le site d'Oracle
- Consultez la section JDK dans l'article dédié à la mise jour de JADE pour plus de détails

## Mise à jour de Ant

Les scripts de Ant sont utitlisés pour construire les livrables. Consultez la section Ant dans l'article dédié à la mise jour de JADE pour plus de détails.

## Mise à jour de Tomcat

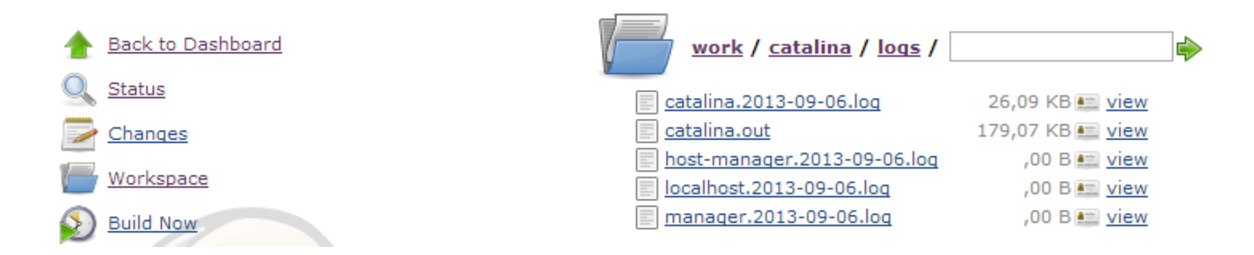
Consultez la section Tomcat dans l'article dédié à la mise jour de JADE pour plus de détails

# 11. FAQ

*Quelle est la méthodologie pour débugger un build ?*

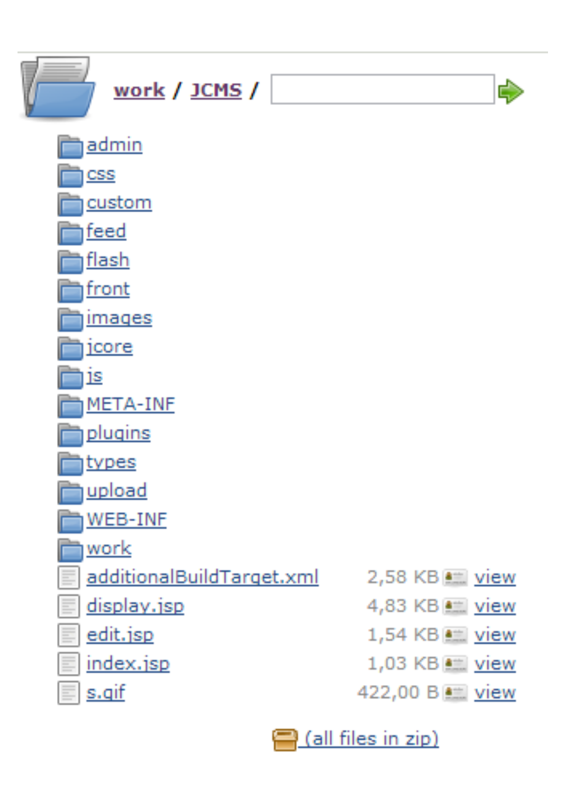
Consulter d'abord le log de build. S'il n'y a pas d'assez d'information :

- Aller dans l'espace de travail du build
- Aller dans `work > catalina > logs > catalina.out`




Si les informations ne sont pas suffisantes, il faudrait débugger directement en lançant le répertoire de travail de JCMS. Il existe deux solutions :

- Soit télécharger le répertoire `work > JCMS` en local puis installer dans Eclipse



- Soit lancer directement l'instance Tomcat du job. Suivre ces étapes pour ce faire :
    - Connecter sur la machine de JADE en ssh
    - Connecter en tant que l'utilisateur « jenkins »
      `$> sudo su -p jenkins`
    - Aller dans le répertoire du job qui ne marche pas
      `$> cd /home/jade/data/jenkins/jobs/MyJob`
    - Indiquer la version tomcat à utiliser
      `$> export CATALINA_HOME=/home/jade/tools/tomcat/tomcat-version`
    - Indiquer le repertoire de travail temporaire du job
      `$> export CATALINA_BASE=/home/jade/data/jenkins/jobs/MyJob/workspace/work/catalina`
    - Lancer
      `$> $CATALINA_HOME/bin/catalina.sh run`
    - Noter le port indiqué à la fin du lancement et ouvrir un navigateur et tester directement. L'URL est normalement : http://jade.company.net:port/jcms

Note : S'il y a un problème pour lancer tomcat, ne pas oublier de positionner le classpath JAVA_HOME
`$> export JAVA_HOME==/home/jade/tools/jdk/jdk-version`

# 12. Terminologie

*Artifact :* ce sont généralement des fichiers comme les modules (.zip) ou la webapp (.war).
*Build job :* ou *job* qui est la configuration d'un projet sous Jenkins. C'est un ensemble de règles (copier un fichier, faire un update de SVN, etc...) qui seront appliquées lors d'un build.
*Build* : une action de construction d'un projet. Un build avec accompagné d'un numéro pour l'identifier. Dans l'environnement de JADE, un build concerne un module ou une webapp. Le résultat d'un build est un ou plusieurs artifacts téléchargeables (.zip, .war)
*ESX* : gamme de produit de la société VMware Inc pour la gestion de virtualisation
*Intégration Continue* : un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée
*VM* : terminologie en anglais pour désigner Virtual Machine
*JDK* : Java Development Kit

